HEH Campus Technique

Nano-ordinateurs

Projet de laboratoire

Titulaires : M. Michiels & D. Arnaud

T. Rosi & A. Urbain
12/05/2017

Table des matières

Présentation générale du projet

Le but du projet est de réaliser un affichage sur la matrice LED 8x8 du Sense HAT des trois axes de rotation du Raspberry Pi : le tangage (axe X), le roulis (axe Y) et le lacet (axe Z).
De plus, il nous était demandé d'afficher la température selon les deux unités principales : le Celsius et le Fahrenheit.
Le Sense HAT comprend plusieurs méthodes prédéfinies permettant de récupérer les valeurs des différents axes ainsi que la température. Le Sense HAT est muni de multiples capteurs et de bibliothèques de méthodes utilisant ceux-ci. C'est grâce à ces bibliothèques de méthodes que nous avons pu réaliser notre projet.

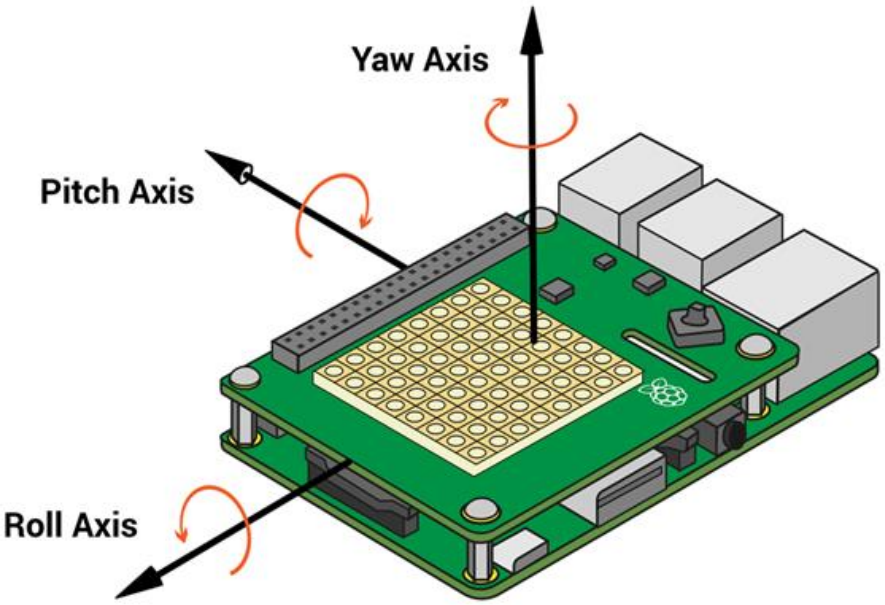

Figure 1 - Axes de rotation du Raspberry Pi

Présentation des capteurs utilisés

Afin de retranscrire de façon numérique son positionnement, le Sense HAT est muni de capteurs environnementaux lui permettant de définir sa position dans l'espace. Pour la réalisation de ce projet, nous avons utilisé les informations récoltées par l'accéléromètre (ST LSM9DS1) via la méthode prédéfinie : get_accelerometer_raw() qui retourne une valeur réelle représentant l'intensité de l'accélération des différents axes en Gs.
Cet accéléromètre mesure l'accélération des forces et peut-être utilisé pour trouver la direction de la gravité par rapport à la position du Sense HAT.

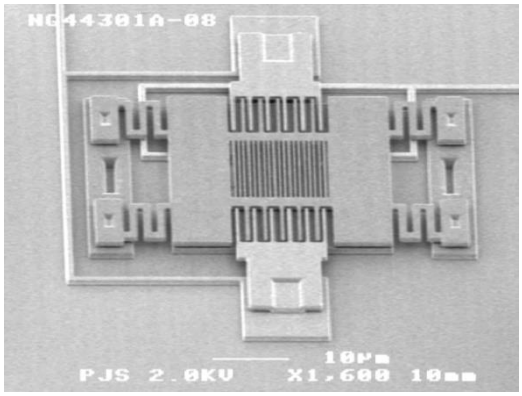

Figure 2 - accéléromètre en nanotechnologie

Sur cette image, nous pouvons apercevoir la masse inertielle située au centre du capteur et maintenue par 4 ressorts. Lorsque cette masse subit une accélération, le support des ressorts va avoir tendance à se déplacer sous l'effet de l'inertie de la masse (celle-ci va rester à sa position initiale) ce qui force le ressort à se comprimer ou à s'étirer ; c'est ce que l'on appelle un *accéléromètre asservi*.
Il existe un autre type d'accéléromètre : le *non asservi* (*piézoélectrique, optique, détection inductive, ...*).
Le principe de l'accéléromètre est basé sur la loi fondamentale de la dynamique :

$$\mathbf{F = m\,a}$$

Lors de sa fabrication, il est placé dans une position de référence (x) représentant le déplacement. Cette même valeur va varier de façon positive ou négative en fonction de l'accélération subie. C'est grâce à la variation de cette valeur que nous pouvons définir les degrés de rotation des différents axes.

Choix de l'IDE et du langage de programmation

Nous avons choisi le langage de programmation Python ainsi que son environnement de développement intégré « IDLE » déjà installé sur le système d'exploitation du Raspberry PI (Linux).
Python est le langage officiel du Raspberry Pi et est installé par défaut dans Raspbian.
Il possède des bibliothèques de méthodes permettant de récupérer les valeurs numériques des différents axes d'orientation.
Le Python est un langage de programmation procédural avec un typage dynamique fort. La facilité d'utilisation et d'apprentissage de ce langage est un de ses points forts, il ne demande pas un énorme bagage de programmation pour être compréhensible. C'est pourquoi le Python s'est imposé à nous.

Algorithme de programmation

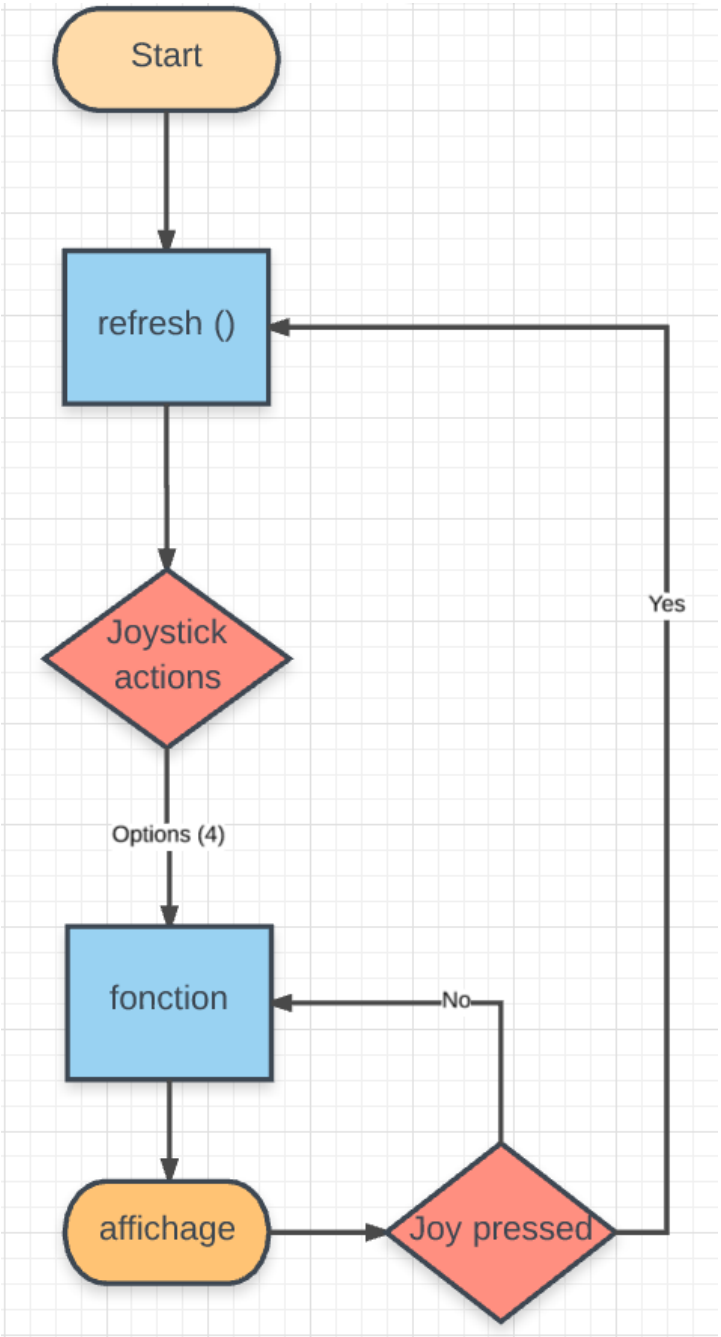

Figure 3 - Représentation schématique de l'algorithme

Notre programme est représenté par 4 fonctions dépendant de l'action sur le joystick (gauche, droite, haut et bas). L'algorithme des 4 fonctions est représenté sur ce diagramme.
Une écoute sur l'évènement ''pressed'' du joystick permet de sortir de l'affichage sélectionné et de retourner au menu principal (refresh).

Problèmes rencontrés

Niveau matériel :

Lors des premiers laboratoires, nous avons passé beaucoup de temps à configurer et à initialiser le Raspberry Pi. En effet, nous avons dû nous y reprendre à plusieurs reprises afin de pouvoir établir une connexion en SSH.
Etant donné cette perte de temps, ainsi que l'indisponibilité du SenseHat en dehors des heures de laboratoire (travail à domicile), nous avons dû prendre quelques heures en dehors de l'horaire prévu afin de finaliser notre projet.
Nous étions conscient de l'existence d'émulateur SenseHat installé sur le Raspberry ainsi que sur internet (Trinket). Malheureusement l'implémentation de la position dans l'espace du Raspberry ne suffisait pas à un résultat cohérent.
La proximité matérielle des câbles d'alimentation et ethernet ainsi que les différents ordinateurs, téléphones, etc… fausse nos résultats lors du calcul d'orientation de l'axe Z.
Les LEDs ont pour défaut de chauffer et donc d'augmenter la température à proximité des capteurs ce qui affecte le résultat affiché.

Niveau technique :

Les premiers résultats de l'orientation de l'axe Z étaient calculés à l'aide de la méthode get_accelerometer_raw(). Ceux-ci ne correspondaient pas à l'orientation de l'axe mais à l'accélération subie par le Raspberry Pi sur l'axe Z. Pour y remédier, nous avons donc utilisé une autre méthode appelée get_orientation_degrees() qui se base sur la position par rapport au nord magnétique. Ceci a ensuite engendré le problème de proximité matérielle cité ci-dessus. Pour y remédier nous avons utilisé un aimant nous permettant de redéfinir artificiellement le nord magnétique.

Programme détaillé

Le code complet (sans coupure) se trouve en annexe A.

```
1   '''Importation des fonctions utiles à la réalisation du projet.'''
2   import numpy as np
3   from sense_hat import SenseHat, ACTION_PRESSED, ACTION_HELD, ACTION_RELEASED
4   from time import sleep
5
```

Figure 4 - Importation des bibliothèques

La 2e ligne importe ‘’numpy ‘’ qui permet de gérer l’affichage à l’aide d’un tableau.
La 3e ligne va importer les différentes méthodes depuis la bibliothèque sense_hat.
La 4e ligne importe la méthode sleep de la bibliothèque time.

```
6   '''Fixe les fonctions de chaque direction du joystick.'''
7   def refresh():
8
9       sense.stick.direction_up = push_up
10      sense.stick.direction_down = push_down
11      sense.stick.direction_left = push_left
12      sense.stick.direction_right = push_right
```

Figure 5 - Méthode faisant office de ''menu''

La méthode refresh (ligne 7) est comparable à un menu, elle écoute les évènements du joystick et appelle d’autres méthodes en fonction du sens dans lequel il est poussé.

```
14  '''Fonction déclanchée lors d'une pression vers la droite du joystick :
15  affiche la température en degrés Celcius.'''
16  def push_right(event):
17      if event.action != ACTION_RELEASED:
18          celsius(sense)
19          sleep(1)
20
21  '''Fonction déclanchée lors d'une pression vers la gauche du joystick :
22  affiche la température en degrés Fahrenheit.'''
23  def push_left(event):
24      if event.action != ACTION_RELEASED:
25          fahrenheit(sense)
26          sleep(1)
27
28  '''Fonction déclanchée lors d'une pression vers le haut du joystick :
29  appelle la fonction display_readings.'''
30  def push_up(event):
31      if event.action != ACTION_RELEASED:
32          display_readings(sense)
33          sleep(1)
34
35  '''Fonction déclanchée lors d'une pression vers le haut du joystick :
36  appelle la fonction display_readings_two.'''
37  def push_down(event):
38      if event.action != ACTION_RELEASED:
39          display_readings_two(sense)
40          sleep(1)
41
```

Figure 6 - 4 méthodes appliquées au Joystick

Chacune des méthodes (push_right, push_left, push_up et push_down) contient un appel à une fonction spécifique qui s'occupera de calculer les valeurs et de les afficher selon un mode défini dans celles-ci.

```
87   def celsius(hat):
88
89           temp = round(sense.get_temperature(), 1)
90
91           sense.show_message("{0}C".format(temp), text_colour=[255, 0, 0])
92           refresh()
93
94   def fahrenheit(hat):
95
96           temp = sense.get_temperature()
97           temp = round((temp * 1.8) + 32, 1)
98
99           sense.show_message("{0}F".format(temp), text_colour=[255, 0, 0])
100
101          refresh()
```

Figure 7 - Méthodes calculant la température (en °C et °F)

Voici les deux méthodes de calcul et d'affichage de la température, d'abord en Celsius (ligne 87) puis en Fahrenheit (ligne 94).

Pour récupérer la température ambiante, nous utilisons une méthode déjà existante :

sense.get_temperature()

que l'on arrondît à un chiffre après la virgule pour éviter un affichage trop long.
La température est ensuite affichée à l'aide de la ligne de commande 99 dans laquelle nous pouvons choisir la couleur d'allumage des LEDs en RGB (rouge, vert et bleu, les couleurs primaires).

Etant donné que la valeur récupérée par la méthode get_temperature() est en Celsius, il faut modifier la valeur pour la passer en Fahrenheit (ligne 97) et ensuite l'afficher comme vu précédemment (ligne 99).

La dernière ligne de chaque méthode renvoie au ''menu principal'' et attend une action sur le joystick.

```
112   '''Permet l'affichage des différents axes d'orientation
113   sur un graphique à bâtonnets.'''
114   def display_readings(hat):
115
116       test = True
117       i = 0
118
119       while test :
120
121           for event in sense.stick.get_events():
122               if "{}".format(event.action) == "pressed" :
123                   test = False
124                   sense.show_message("Choice", text_colour=[255, 0, 0])
125                   refresh()
126
127
128           n = (0, 0, 0)
129           r = (255, 0, 0)
130           g = (0, 255, 0)
131           b = (0, 0, 255)
132
133           acceleration = sense.get_accelerometer_raw()
134           x = acceleration['x']
135           y = acceleration['y']
136
137           orientation = sense.get_orientation_degrees()
138           lacet = orientation['yaw']
139
140           value_X = evaluate(x*180)
141           value_Y = evaluate(y*180)
142           value_Z = evaluate(lacet-180)
143
144           i = i+1
145           if(i==30) :
146               i = 0
147
148               screen = [  n, n, n, n, n, n, n, n,
149                           n, n, n, n, n, n, n, n,
150                           n, n, n, n, n, n, n, n,
151                           n, n, n, n, n, n, n, n,
152                           n, n, n, n, n, n, n, n,
153                           n, n, n, n, n, n, n, n,
154                           n, n, n, n, n, n, n, n,
155                           n, n, n, n, n, n, n, n]
```

Figure 8 - Méthode pour le premier affichage

Nous avons ci-dessus une des deux méthodes qui affiche un graphe représentant l'orientation du Raspberry Pi dans l'espace (voir Annexes, Figure – 14).

Tout d'abord, la variable ***test*** (ligne 116) est un booléen qui va nous permettre d'exécuter et d'interrompre notre boucle sous certaines conditions.
Le ***i = 0*** sera utilisé pour laisser la boucle while (ligne 119) tourner 30 fois avant d'afficher le résultat sur les LEDs. On incrémente ***i*** à chaque passage de boucle (ligne 144).

Les lignes 121 à 125 permettent à chaque début de boucle d'écouter si le joystick est pressé, ce qui dans ce cas permet de quitter la méthode et de retourner au ''menu principal''.

Les lignes 128 à 131 créent 4 variables définissant chacune une couleur spécifique qui seront utilisées lors de l'affichage.
Les lignes 133 et 137 sont les plus importantes, ce sont elles qui récupèrent les valeurs de positionnement du Raspberry Pi et qui les stockent dans ***acceleration et orientation***.

La ligne 134 récupère la valeur de la case 'x' du tableau associatif acceleration et la stocke dans la variable x (idem pour ligne 135 et 138).
De 140 à 142, les valeurs précédentes sont envoyées à la méthode ***evaluate*** (qui sera décrite plus tard), le résultat est stocké dans différentes variables qui représentent la valeur de chaque axe.

A partir de 145 jusque 174, la condition d'entrée est ***i = 30*** c'est-à-dire que toute cette section ne sera exécutée que lorsque la valeur de ***i*** sera égale à 30 (donc après 30 passage de boucle vu que ***i*** est incrémenté en ligne 144). Cette condition est utilisée pour laisser le temps au Raspberry d'effectuer les calculs et d'ensuite afficher les résultats. L'affichage étant une portion de code assez lourde à exécuter, nous avions au départ (sans la condition) un affichage des valeurs qui prenait beaucoup trop de temps à se stabiliser, d'où l'existence de cette condition.
On crée un tableau ***screen*** remplis de ***n*** (noir) qui sera modifié en fonction des valeurs reçue depuis ***evaluate***.

```
147             if value_X >= 4 :
148                 for i in range (4, value_X+1):
149                     screen[i] = r
150                     screen[8+i] = r
151             else :
152                 for i in range (value_X, 4):
153                     screen[i] = r
154                     screen[8+i] = r
155
156             if value_Y >= 4 :
157                 for i in range (4, value_Y+1):
158                     screen[24+i] = g
159                     screen[32+i] = g
160             else :
161                 for i in range (value_Y, 4):
162                     screen[24+i] = g
163                     screen[32+i] = g
164
165             if value_Z >= 4 :
166                 for i in range (4, value_Z+1):
167                     screen[48+i] = b
168                     screen[56+i] = b
169             else :
170                 for i in range (value_Z, 4):
171                     screen[48+i] = b
172                     screen[56+i] = b
173
174             sense.set_pixels(screen)
175
```

Figure 9 – Suite et fin de la figure 8

Nous avons ici plusieurs structures conditionnelles *if / else* que nous lisons : *si* la valeur de x (ligne 147) est plus grande ou égale à 4, tu fais ceci (ligne 148 à 150), *sinon* (ligne 151) tu fais cela (ligne 152 à 154).

Ces structures se répètent pour chaque variable (value_X, value_Y et value_Z) et modifient la couleur à afficher à une position bien précise.

En ligne 174, le tableau modifié sera affiché sur la matrice LEDs 8x8 du Sense HAT.

```
42  '''Evalue le résultat de l'orientation du SenseHat en degré et retourne
43  une valeur comprise entre 0 et 7 permettant un intervalle correct facilitant
44  l'affichage sur la matrice 8x8 des LEDs du SenseHat.'''
45  def evaluate (axe):
46      val = 0
47      if axe > 0 :
48          if axe < 45:
49              val = 4
50          elif axe < 90:
51              val = 5
52          elif axe < 135:
53              val = 6
54          else :
55              val = 7
56      else :
57          if axe > -45:
58              val = 3
59          elif axe > -90:
60              val = 2
61          elif axe > -135:
62              val = 1
63          else :
64              val = 0
65      return val
66
```

Figure 10 - Méthode ''evaluate''

Cette méthode reçoit une valeur en paramètre qui permet de retourner un entier compris entre 0 et 7, correspondant à la matrice LEDs 8x8 du Sense HAT.

```
186   '''Permet un affichage des différents axes d'orientation.
187   Les axes X et Y sont représentés par une croix et l'axe Z
188   par un point représentant le nord.'''
189   def display_readings_two(hat):
190
191       test = True
192       i = 0
193
194       while test :
195
196           for event in sense.stick.get_events():
197               if "{}".format(event.action) == "pressed" :
198                   test = False
199                   sense.show_message("Choice", text_colour=[255, 0, 0])
200                   refresh()
201
202
203           n = (0, 0, 0)
204           r = (255, 0, 0)
205           g = (0, 255, 0)
206           b = (0, 0, 255)
207           w = (150,150,150)
208
209           acceleration = sense.get_accelerometer_raw()
210           x = acceleration['x']
211           y = acceleration['y']
212
213           orientation = sense.get_orientation_degrees()
214           lacet = orientation['yaw']
215
216           value_X = evaluate(x*180)
217           value_Y = evaluate(y*180)
218           value_Z = evaluate_two(lacet+180)
219
220           i = i+1
221           if(i==30) :
222               i = 0
223
224               screen = [  n, n, n, n, n, n, n, n,
225                           n, n, n, n, n, n, n, n,
226                           n, n, n, n, n, n, n, n,
227                           n, n, n, w, w, n, n, n,
228                           n, n, n, w, w, n, n, n,
229                           n, n, n, n, n, n, n, n,
230                           n, n, n, n, n, n, n, n,
231                           n, n, n, n, n, n, n, n]
```

Figure 11 - Méthode pour le second affichage

```
221             if value_Y >= 4 :
222                 screen[43] = g
223                 screen[44] = g
224                 if value_Y >= 6 :
225                     screen[51] = g
226                     screen[52] = g
227             elif value_Y <= 3 :
228                 screen[19] = g
229                 screen[20] = g
230                 if value_Y <= 1 :
231                     screen[11] = g
232                     screen[12] = g
233
234             if value_X >= 4 :
235                 screen[29] = r
236                 screen[37] = r
237                 if value_X >= 6 :
238                     screen[30] = r
239                     screen[38] = r
240             elif value_X <= 3 :
241                 screen[26] = r
242                 screen[34] = r
243                 if value_X <= 1 :
244                     screen[25] = r
245                     screen[33] = r
246
247             if value_Z == 0 :
248                 screen[16] = b
249                 screen[24] = b
250                 screen[32] = b
251                 screen[40] = b
252             elif value_Z == 1 :
253                 screen[0] = b
254                 screen[1] = b
255                 screen[8] = b
256             elif value_Z == 2 :
257                 screen[2] = b
258                 screen[3] = b
259                 screen[4] = b
260                 screen[5] = b
261             elif value_Z == 3 :
262                 screen[6] = b
263                 screen[7] = b
264                 screen[15] = b
265             elif value_Z == 4 :
266                 screen[23] = b
267                 screen[31] = b
268                 screen[39] = b
269                 screen[47] = b
270             elif value_Z == 5 :
271                 screen[55] = b
272                 screen[62] = b
273                 screen[63] = b
274             elif value_Z == 6 :
275                 screen[58] = b
276                 screen[59] = b
277                 screen[60] = b
278                 screen[61] = b
279             elif value_Z == 7 :
280                 screen[48] = b
281                 screen[56] = b
282                 screen[57] = b
283
284             sense.set_pixels(screen)
285
```

Figure 12 - suite figure 11

```
67  '''Evalue le résultat de l'orientation du SenseHat en degré et retourne
68  une valeur comprise entre 0 et 7 permettant un intervalle correct facilitant
69  l'affichage sur la matrice 8x8 des LEDs du SenseHat
70  (Pour le deuxième mode d'affichage).'''
71  def evaluate_two(axe):
72      if axe >= 360:
73          axe = axe - 360
74      val = 0
75      if axe <= 22 :
76          val = 0
77      elif axe <= 67 :
78          val = 7
79      elif axe <= 112 :
80          val = 6
81      elif axe <= 157 :
82          val = 5
83      elif axe <= 202 :
84          val = 4
85      elif axe <= 247 :
86          val = 3
87      elif axe <= 292 :
88          val = 2
89      elif axe <= 337 :
90          val = 1
91      else :
92          val = 0
93      return val
```

Figure 13 - Méthode "evaluate_two"

Ces portions de code sont semblables aux précédentes, c'est uniquement l'ordre des LEDs et le calcul des valeurs dans ***evaluate_two*** qui varient légèrement pour s'adapter au format d'affichage (voir Annexes, Figure – 15).

```
297  sense = SenseHat()
298  sense.clear()
299
300  '''Boucle infinie permettant au programme d'être
301  utilisé de façon continue.'''
302  while True :
303      refresh()
304      sleep(1)
```

Création d'un objet Sense HAT (ligne 297). Nettoyage de la matrice LEDs (ligne 298). Structure de contrôle permettant d'exécuter les instructions du programme de manière infinie (ligne 302).

Conclusion

Durant ce projet, nous avons appris à configurer un Raspberry Pi et à établir une connexion SSH afin de travailler sur celui-ci à partir d'un ordinateur connecté sur le même réseau. Nous nous sommes très vite familiarisé avec l'interface utilisateur du Raspberry qui est très intuitive et facile d'accès, même pour des débutants en Linux. Nous avons découvert et utilisé le langage Python, ce qui a enrichi notre bagage en programmation.

Nous avons appris à utiliser le Sense HAT et à tirer profit des capteurs qui le compose. De par nos recherches nous avons pu constater toutes les possibilités qu'offre un nano-ordinateur.
De plus, ce projet en binôme nous a permis de pallier aux faiblesses de chacun en s'appuyant sur nos propres compétences.

Pour aller plus loin, l'affichage peut être amélioré grâce à des méthodes plus complexes comme par exemple :

- L'affichage des valeurs d'orientation en numérique ;
- L'affichage sur un graphique a plus petite échelle pour améliorer la précision (sur écran externe) ;
- Améliorer le code pour le rendre plus modulable (permettre la réutilisation dans un autre projet) ;
- Afficher l'humidité et la pression atmosphérique ;
- Pour l'affichage de la température, utiliser le gyroscope pour orienter le texte en fonction de la position du Raspberry Pi

Bibliographie

- François MOCQ, ‘’Sense HAT, un tour dans les étoiles…’’, sur http://www.framboise314.fr/sense-hat-un-tour-dans-les-etoiles/, consulté le 09/05/2017, (F. MOCQ se présente comme informaticien et créateur du blog).

- STMicroelectronics NV, ‘’ iNEMO inertial module : 3D accelerometer, 3D gyroscope, 3D magnetometer’’, [PDF], 2015. URL : http://www.st.com/content/ccc/resource/technical/document/datasheet/1e/3f/2a/d6/25/eb/48/46/DM00103319.pdf/files/DM00103319.pdf/jcr:content/translations/en.DM00103319.pdf

- Raspberry Pi Learning Ressources, ‘’Getting Started with the Sense Hat’’, sur https://www.raspberrypi.org/learning/getting-started-with-the-sense-hat/worksheet/, consulté le 09/05/2017.

- Raspberry Pi Learning Ressources, ‘’Movement’’, sur https://www.raspberrypi.org/learning/astro-pi-guide/sensors/movement.md, consulté le 21/02/2017.

- Figure 1 : Framboise314.fr, ‘’pitch_yaw_roll_sensehat’’, [png], http://www.framboise314.fr/wp-content/uploads/2015/09/pitch_yaw_roll_sensehat.png, consulté le 09/05/2017.

- Figure 2 : Framboise314.fr, ‘’accelerometre ‘’, [png], http://www.framboise314.fr/wp-content/uploads/2015/09/accelerometre.jpg, consulté le 09/05/2017.

HEH Campus Technique

Nano-ordinateurs

Annexes

Projet de laboratoire

Titulaires : M. Michiels & D. Arnaud

T. Rosi & A. Urbain
12/05/2017

Annexe A

```
'''Importation des fonctions utiles à la réalisation du projet.'''
import numpy as np
from sense_hat import SenseHat, ACTION_PRESSED, ACTION_HELD, ACTION_RELEASED
from time import sleep

'''Fixe les fonctions de chaque direction du joystick.'''
def refresh():

    sense.stick.direction_up = push_up
    sense.stick.direction_down = push_down
    sense.stick.direction_left = push_left
    sense.stick.direction_right = push_right

'''Fonction déclanchée lors d'une pression vers la droite du joystick :
affiche la température en degrés Celcius.'''
def push_right(event):
    if event.action != ACTION_RELEASED:
        celsius(sense)
        sleep(1)

'''Fonction déclanchée lors d'une pression vers la gauche du joystick :
affiche la température en degrés Fahrenheit.'''
def push_left(event):
    if event.action != ACTION_RELEASED:
        fahrenheit(sense)
        sleep(1)

'''Fonction déclanchée lors d'une pression vers le haut du joystick : appelle
la fonction display_readings.'''
def push_up(event):
    if event.action != ACTION_RELEASED:
        display_readings(sense)
        sleep(1)

'''Fonction déclanchée lors d'une pression vers le haut du joystick : appelle
la fonction display_readings_two.'''
def push_down(event):
    if event.action != ACTION_RELEASED:
        display_readings_two(sense)
        sleep(1)
```

```
'''Evalue le résultat de l'orientation du SenseHat en degré et retourne une
valeur comprise entre 0 et 7 permettant un intervalle correct facilitant
l'affichage sur la matrice 8x8 des LEDs du SenseHat.'''
def evaluate (axe):
    val = 0
    if axe > 0 :
        if axe < 45:
            val = 4
        elif axe < 90:
            val = 5
        elif axe < 135:
            val = 6
        else :
            val = 7
    else :
        if axe > -45:
            val = 3
        elif axe > -90:
            val = 2
        elif axe > -135:
            val = 1
        else :
            val = 0
    return val
```

```
'''Evalue le résultat de l'orientation du SenseHat en degré et retourne une valeur comprise entre 0 et 7 permettant un intervalle correct facilitant l'affichage sur la matrice 8x8 des LEDs du SenseHat (Pour le deuxième mode d'affichage).'''
def evaluate_two(axe):
    if axe >= 360:
        axe = axe - 360
    val = 0
    if axe <= 22 :
        val = 0
    elif axe <= 67 :
        val = 7
    elif axe <= 112 :
        val = 6
    elif axe <= 157 :
        val = 5
    elif axe <= 202 :
        val = 4
    elif axe <= 247 :
        val = 3
    elif axe <= 292 :
        val = 2
    elif axe <= 337 :
        val = 1
    else :
        val = 0
    return val


def celsius(hat):

        temp = round(sense.get_temperature(), 1)

        sense.show_message("{0}C".format(temp), text_colour=[255, 0, 0])
        refresh()

def fahrenheit(hat):

        temp = sense.get_temperature()
        temp = round((temp * 1.8) + 32, 1)

        sense.show_message("{0}F".format(temp), text_colour=[255, 0, 0])
        refresh()
```

```
'''Permet l'affichage des différents axes d'orientation sur un graphique à
bâtonnets.'''
def display_readings(hat):

    test = True
    i = 0

    while test :

        for event in sense.stick.get_events():
            if "{}".format(event.action) == "pressed" :
                test = False
                sense.show_message("Choice", text_colour=[255, 0, 0])
                refresh()


        n = (0, 0, 0)
        r = (255, 0, 0)
        g = (0, 255, 0)
        b = (0, 0, 255)

        acceleration = sense.get_accelerometer_raw()
        x = acceleration['x']
        y = acceleration['y']

        orientation = sense.get_orientation_degrees()
        lacet = orientation['yaw']

        value_X = evaluate(x*180)
        value_Y = evaluate(y*180)
        value_Z = evaluate(lacet-180)

        i = i+1
        if(i==30) :
            i = 0

            screen = [  n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n]
```

```
if value_X >= 4 :
    for i in range (4, value_X+1):
        screen[i] = r
        screen[8+i] = r
else :
    for i in range (value_X, 4):
        screen[i] = r
        screen[8+i] = r

if value_Y >= 4 :
    for i in range (4, value_Y+1):
        screen[24+i] = g
        screen[32+i] = g
else :
    for i in range (value_Y, 4):
        screen[24+i] = g
        screen[32+i] = g

if value_Z >= 4 :
    for i in range (4, value_Z+1):
        screen[48+i] = b
        screen[56+i] = b
else :
    for i in range (value_Z, 4):
        screen[48+i] = b
        screen[56+i] = b

sense.set_pixels(screen)
```

```
'''Permet un affichage des différents axes d'orientation. Les axes X et Y sont représentés par une croix et l'axe Z par un point représentant le nord.'''
def display_readings_two(hat):

    test = True
    i = 0

    while test :

        for event in sense.stick.get_events():
            if "{}".format(event.action) == "pressed" :
                test = False
                sense.show_message("Choice", text_colour=[255, 0, 0])
                refresh()


        n = (0, 0, 0)
        r = (255, 0, 0)
        g = (0, 255, 0)
        b = (0, 0, 255)
        w = (150,150,150)

        acceleration = sense.get_accelerometer_raw()
        x = acceleration['x']
        y = acceleration['y']

        orientation = sense.get_orientation_degrees()
        lacet = orientation['yaw']

        value_X = evaluate(x*180)
        value_Y = evaluate(y*180)
        value_Z = evaluate_two(lacet+180)

        i = i+1
        if(i==30) :
            i = 0

            screen = [  n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, w, w, n, n, n,
                        n, n, n, w, w, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n,
                        n, n, n, n, n, n, n, n]
```

```
if value_Y >= 4 :
    screen[43] = g
    screen[44] = g
    if value_Y >= 6 :
        screen[51] = g
        screen[52] = g
elif value_Y <= 3 :
    screen[19] = g
    screen[20] = g
    if value_Y <= 1 :
        screen[11] = g
        screen[12] = g

if value_X >= 4 :
    screen[29] = r
    screen[37] = r
    if value_X >= 6 :
        screen[30] = r
        screen[38] = r
elif value_X <= 3 :
    screen[26] = r
    screen[34] = r
    if value_X <= 1 :
        screen[25] = r
        screen[33] = r

if value_Z == 0 :
    screen[16] = b
    screen[24] = b
    screen[32] = b
    screen[40] = b
elif value_Z == 1 :
    screen[0] = b
    screen[1] = b
    screen[8] = b
elif value_Z == 2 :
    screen[2] = b
    screen[3] = b
    screen[4] = b
    screen[5] = b
elif value_Z == 3 :
    screen[6] = b
    screen[7] = b
    screen[15] = b
```

```
            elif value_Z == 4 :
                screen[23] = b
                screen[31] = b
                screen[39] = b
                screen[47] = b
            elif value_Z == 5 :
                screen[55] = b
                screen[62] = b
                screen[63] = b
            elif value_Z == 6 :
                screen[58] = b
                screen[59] = b
                screen[60] = b
                screen[61] = b
            elif value_Z == 7 :
                screen[48] = b
                screen[56] = b
                screen[57] = b

            sense.set_pixels(screen)

sense = SenseHat()
sense.clear()

'''Boucle infinie permettant au programme d'être utilisé de façon continue.'''
while True :
    refresh()
    sleep(1)
```

Annexe B

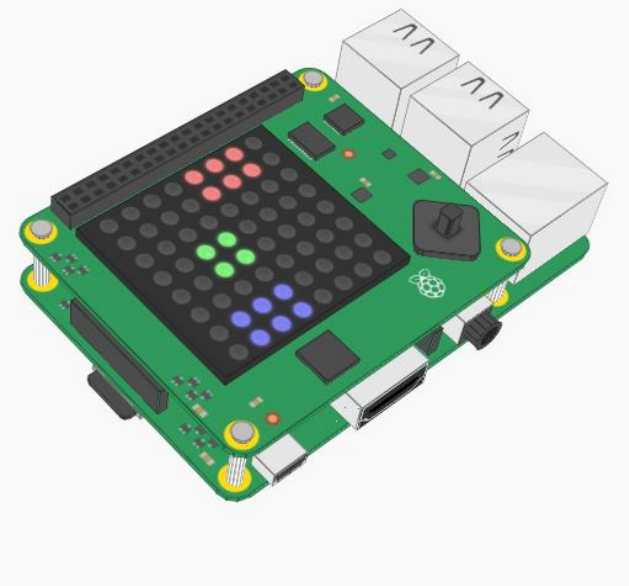

Figure 15 - Affichage 1

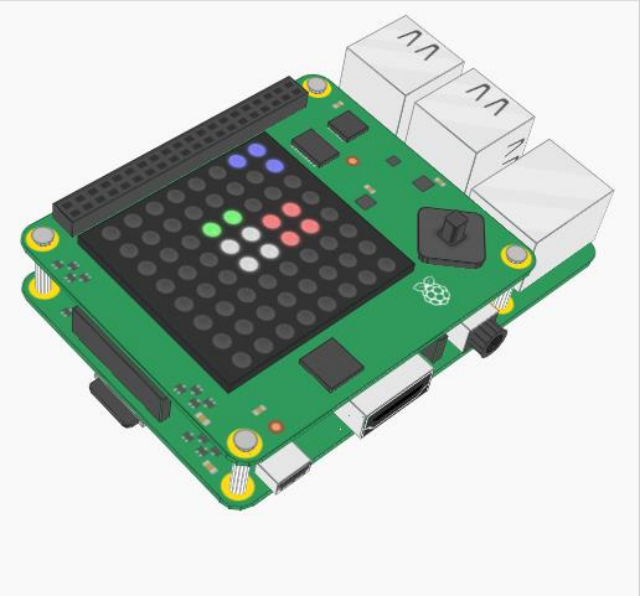

Figure 14 - Affichage 2